Curso de Formação LPIC-1

Exame 101




Bruno Santos <brunosantos@whatgeek.com.pt>

Curso Linux: formação

- Gestão de Processos
  - Processos
  - Sinais
  - Prioridades
  - Controlo de trabalhos

Processos

Processos

- Gerir processos é essencial numa máquina Linux
- Gestão e controlo dos processos é importante
- Maioria dos casos, geridos pelo kernel
- Re-leitura dos ficheiros de configuração

- Processos, o que são?
  - Qualquer programa
    - Seja um comando
    - Aplicação
    - script

Processos

Processos: Atributos e conceitos

- Tempo de vida
  - Definido pelo tempo que dura a sua execução
    - ls
    - browsers (terminados manualmente)
    - daemons de servidores – correm continuamente
  - Definido pelo tempo que dura a sua execução
- Process ID (PID)
  - Todos os processos têm um número associado quando iniciam. PID's são números inteiros únicos
- User ID (UID) e Group ID (GID)
  - Processos têm privilégios associados, e o UID e GID estão associados ao utilizador que começou o processo. Isto limita o acesso do processo a objectos do sistema.

Processos

Processos: Atributos e conceitos

- Processo pai
  - O primeiro processo iniciado pelo kernel no arranque do sistema é um programa chamado *init*. Este processo tem o PID 1 e é o pai de todos os processos restantes do sistema. A nossa *shell* é um descendente do *init* e o processo pai de todos os comandos iniciados pela *shell*, que são processos filho ou sub-processos.
- ID processo pai (Parent ID)
  - Este é o PID do processo que iniciou o processo em questão. Se o processo pai desaparecer, o PID irá ser 1, que é o PID do *init*.
- Ambiente
  - Cada processo contém uma lista de variáveis e valores associados. Colectivamente, esta lista é conhecida como o ambiente do processo, e as varáveis chamadas de variáveis de ambiente. Processos filhos herdam o seu ambiente do processo pai, excepto quando outro ambiente é especificado quando o programa é executado.

Processos

Processos: Atributos e conceitos

- Directoria corrente de trabalho
  - A directoria por defeito associada a cada processo. O processo irá ler e escrever ficheiros nesta directoria, excepto se explicitamente for especificada outra no sistema

Processos

Processos: Estado dos processos

- Um processo não é automaticamente elegível para receber tempo de processamento só porque existe.
- Quatro estados de execução:
  - Runnable – processo pode ser executado
  - Sleeping – processo está à espera de algum recurso
  - Zombie – processo está a tentar “morrer”
  - Stopped – processo está suspenso (não lhe é permitido executar)

- Runnable – está pronto para executar quando houver tempo de processamento disponível. Adquiriu todos os recursos que precisa e só espera por tempo de processamento.
- Sleeping – está á espera que algum evento especifico ocorra. O processo é bloqueado até que o seu pedido seja satisfeito, não obtém qualquer tempo de processamento.
- Zombies – processos que terminaram a sua execução, mas ainda não viram o seu estado actualizado.
- Stopped – Administrativamente proibidos de executar. Similar a *Sleeping* mas precisam que seja outro processo a “acordá-los” ou a “matá-los”

Processos

Processos: Monitorização

- A qualquer altura podem estar dezenas ou mesmo centenas de processos em execução num sistema Linux
- Comandos como ps, pstree e top ajudam-nos a visualizar este complexo sistema.

```
root     4817  4814  0 Jul11 ?       00:00:01 hald-addon-input: Listening on /dev/input/event3
/dev/input/event2 /dev/input/event1 /dev/in
102      4823  4814  0 Jul11 ?       00:00:00 hald-addon-acpi: listening on acpid socket
/var/run/acpid.socket
root     4824  4814  0 Jul11 ?       00:00:10 hald-addon-storage: polling /dev/sdf (every 2 sec)
root     4826  4814  0 Jul11 ?       00:00:18 hald-addon-storage: polling /dev/sdb (every 2 sec)
root     4843  4814  0 Jul11 ?       00:00:56 hald-addon-storage: polling /dev/sr0 (every 2 sec)
bin      4910     1  0 Jul11 ?       00:00:00 /sbin/portmap
root     5056     1  0 Jul11 ?       00:00:00 /usr/kde/3.5/bin/kdm
root     5085  5056  0 Jul11 tty7    00:12:06 /usr/bin/X -br -nolisten tcp :0 vt7 -auth /var/run/xauth/A:0-
RyYGbp
root     5113     1  0 Jul11 ?       00:00:00 /usr/sbin/sshd
root     5309     1  0 Jul11 ?       00:00:00 /usr/sbin/cupsd
root     5368     1  0 Jul11 ?       00:00:09 /sbin/dhcdbd --system
root     5425     1  0 Jul11 ?       00:00:00 /usr/sbin/NetworkManager
root     5507     1  0 Jul11 ?       00:00:00 /usr/sbin/cron
root     5570     1  0 Jul11 tty1    00:00:00 /sbin/agetty 38400 tty1 linux
1000     5612  5196  0 Jul11 ?       00:00:00 /bin/sh /usr/kde/3.5/bin/startkde
1000     5649     1  0 Jul11 ?       00:00:00 /usr/bin/dbus-launch --sh-syntax --exit-with-session
```

Processos

Processos: ps

- Ps
  - Ferramenta primária de qualquer administrador de sistemas para monitorizar processos.
  - Mostra informações como:
    - PID
    - UID
    - Prioridade
    - CPU
    - Estado
    - (...)
  - Comportamento tende a variar pelas várias versões de UNIX
  - Várias implementações
  - Tornou-se muito complexo

Processo único do utilizador

```
feiticeir0@prometheus ~ $ ps
  PID TTY          TIME CMD
23794 pts/2    00:00:00 bash
23891 pts/2    00:00:00 ps
```

Processos

Processos: ps

```
ps -a
  PID TTY          TIME CMD
24197 pts/2    00:00:00 ps
```

```
ps a
  PID TTY      STAT   TIME COMMAND
 5085 tty7     Ss+   12:16 /usr/bin/X -br -nolisten tcp :0 vt7 -auth
/var/run/xauth/A:0-RyYGbp
 5570 tty1     Ss+    0:00 /sbin/agetty 38400 tty1 linux
 5571 tty2     Ss+    0:00 /sbin/agetty 38400 tty2 linux
 5572 tty3     Ss+    0:00 /sbin/agetty 38400 tty3 linux
 5573 tty4     Ss+    0:00 /sbin/agetty 38400 tty4 linux
 5574 tty5     Ss+    0:00 /sbin/agetty 38400 tty5 linux
 5577 tty6     Ss+    0:00 /sbin/agetty 38400 tty6 linux
 5710 pts/1    Ss+    0:00 /bin/bash
23794 pts/2    Ss     0:00 /bin/bash
24208 pts/2    R+     0:00 ps a
```

Processos

Processos: ps

- Ps
  - a : mostra processos de todos os utilizadores
  - u : mostra informação do utilizador para os processos
  - x : mostra processos sem um tty a controlar

```
ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   3764   300 ?        Ss   Jul11   0:02 init [3]
root         2  0.0  0.0      0     0 ?        S<   Jul11   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S<   Jul11   0:00 [migration/0]
root         4  0.0  0.0      0     0 ?        S<   Jul11   0:01 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   Jul11   0:00 [migration/1]
root         6  0.0  0.0      0     0 ?        S<   Jul11   0:05 [ksoftirqd/1]
root         7  0.0  0.0      0     0 ?        S<   Jul11   0:00 [cpuset]
root         8  0.0  0.0      0     0 ?        S<   Jul11   0:00 [events/0]
root         9  0.0  0.0      0     0 ?        S<   Jul11   0:01 [events/1]
root        10  0.0  0.0      0     0 ?        S<   Jul11   0:00 [work_on_cpu/0]
root        11  0.0  0.0      0     0 ?        S<   Jul11   0:00 [work_on_cpu/1]
root      5113  0.0  0.0  37792   440 ?        Ss   Jul11   0:00 /usr/sbin/sshd
root      5196  0.0  0.0  55156   388 ?        S    Jul11   0:00 -:0
1000      5665  0.0  0.1 120648  2880 ?        Ss   Jul11   0:01 kdeinit Running...
1000      5668  0.0  0.1 121528  2628 ?        S    Jul11   0:04 dcopserver [kdeinit] --nosid
1000      5670  0.0  0.1 125244  3836 ?        S    Jul11   0:02 klauncher [kdeinit] --new-startup
1000      5672  0.0  0.4 177636  8696 ?        S    Jul11   0:09 kded [kdeinit] --new-startup
1000      5674  0.0  0.0  15448  1000 ?        S    Jul11   0:01 /usr/libexec/gam_server
1000      5679  0.0  0.0   3740   140 ?        S    Jul11   0:04 kwrapper ksmserver
```

Processos

Processos: ps

Campo	Conteúdo
USER	Username do dono do processo
PID	ID do processo (Process ID)
%CPU	% do cpu usada por este processo
%MEM	% de memória usada pelo processo
VSZ	Tamanho virtual do processo
RSS	N.º de páginas em memória (Resident set size)
TTY	ID do terminal de controlo
STAT	Status corrente do processo R=runnable D=”dormir para sempre” S=sleeping (<20s) T=traçados ou parados Z=zombie
START	Tempo de inicio do processo
TIME	Tempo de CPU consumido pelo processo
Comand	Nome do comando e argumentos

Processos

Processos: pstree

- pstree
  - Mostra um esquema hierárquico dos processos no sistema

```
init─┬─NetworkManager───{NetworkManager}
     ├─acpid
     ├─6*[agetty]
     ├─amarokapp─┬─ruby
     │           └─6*[{amarokapp}]
     ├─apache2─┬─apache2
     │         └─2*[apache2───26*[{apache2}]]
     ├─cron
     ├─dhcdbd───dhclient
     ├─gam_server
     ├─gnome-keyring-d
     ├─hald───hald-runner─┬─hald-addon-acpi
     │                    ├─hald-addon-inpu
     │                    └─7*[hald-addon-stor]
     ├─hcid
     ├─kded
     ├─kdeinit─┬─firefox───run-mozilla.sh───firefox-bin───5*[{firefox-bin}]
     │         ├─kio_file
     │         ├─konqueror
     │         ├─konsole───bash───pstree
     │         └─wish───{wish}
     ├─kdesktop
```

Processos

Processos: sinais

- Sinais são interrupções ao nível dos processos
- Cerca de 30 diferentes estão definidos
- São usados para variadas causas:
  - Enviados entre processos como forma de comunicação
  - Enviados pelo terminal para “matar”, interromper ou suspender processos
  - Enviados pelo administrador (com o kill) para alcançar vários resultados
  - Enviados pelo kernel quando um processo comete uma infracção (ex: divisão por zero)
  - Enviados pelo kernel para notificar um processo de uma “condição especial”

Processos

Processos: sinais

#	Nome	Descrição	Processo
1	HUP	Hangup	Termina
2	INT	Interrupt	Termina
3	QUIT	Quit	Termina
9	KILL	Kill	Termina
11	SEGV	Segmentation Fault	Termina
15	TERM	Software Termination	Terminate

Processos

Processos: sinais

- Sinais são interrupções ao nível dos processos
- Cerca de 60 diferentes estão definidos
- São usados para variadas causas:
  - Enviados entre processos como forma de comunicação
  - Enviados pelo terminal para “matar”, interromper ou suspender processos
  - Enviados pelo administrador (com o kill) para alcançar vários resultados
  - Enviados pelo kernel quando um processo comete uma infracção (ex: divisão por zero)
  - Enviados pelo kernel para notificar um processo de uma “condição especial”

Processos

Processos: kill e killall

- Método normal de parar um processo:
  - Kill
  - Killall
  - Enviar um pedido de “por favor” ao processo
- Apenas enviam um sinal ao processo. Não o removem da memória
- O sinal por defeito é o número 15 (SIGTERM)
  - Pede “por favor” ao processo para terminar
- O sinal SIGUP ou HUP (número 1) é um caso especial
  - Processo não é “morto” mas assinalado para se auto-terminar e reiniciar.
  - Uso mais frequente é para ler novamente os ficheiros de configuração
- O sinal SIGKILL (número 9) é o sinal mais “forte”
  - Mata inclusive processos zombie
  - Mata os processos e remove-os da memória
  - Um processo morto desta forma não guarda nenhum dado

Processos

Processos: kill e killall

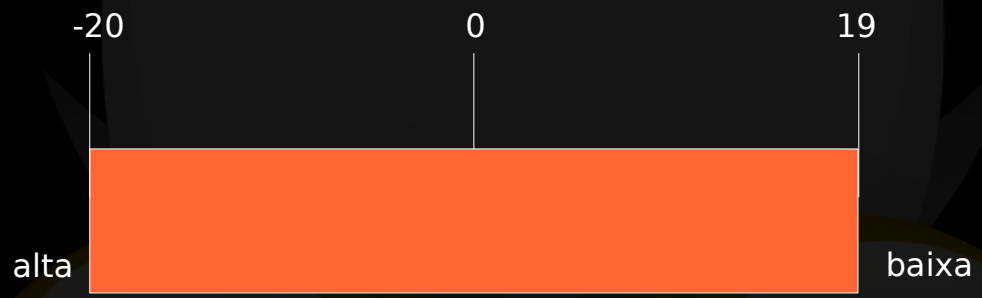
- Abrir uma shell
- Abrir outra shell
  - ps u
  - descobrir qual a shell mais recente. Retirar o PID
  - kill -9 PID ou kill -KILL PID

- Killall – mata processos pelo nome
- Abrir uma shell
  - Executar top
- Abrir outra shell
  - Executar top
- Abrir outra shell
  - killall -TERM top

Processos

Processos: Prioridades

- Linux usa uma combinação de prioridades e escalonamento para correr vários processos
- Estranhamente, as prioridades dos processos são descendentes
- Vão desde -20 (mais alta) até 19 (mais baixa)
- A prioridade por defeito de um processo, normalmente é 0, excepto quando precisa de definir uma mais alta
- Utilizadores não podem baixar – 0 a 19
- Apenas o root pode aumentar a prioridade – 0 a -20



Processos

Processos: Prioridades

- NUNCA COLOCAR MUITOS PROCESSOS COM PRIORIDADES ELEVADAS
- É DESASTRE CERTO
- Duas formas principais de alterar prioridades num processo:
  - Ao iniciar o programa
  - Durante a sua execução
- Ao usar o programa nice, o processo começa com prioridade 10

```
time nice dd if=/dev/zero of=ficheiro bs=1M count=1024
```

```
time dd if=/dev/zero of=ficheiro bs=1M count=1024
```

```
dd if=/dev/zero of=ficheiro2 bs=1M count=1024
```

Processos

Processos: Prioridades

- Quando um processo está em execução, usamos o programa renice
- renice +<valor> PID
- UTILIZADOR NÃO PODE AUMENTAR PRIORIDADES

```
renice +5 23711
23711: old priority 0, new priority 5
```

Processos

Processos: top

- Mostra uma listagem de todos os processos com maior utilização do CPU
- Configurável para mostrar apenas processos activos ou um número especifico de processos
- top corre interactivamente, refrescando a página de 5s em 5s

```
top - 19:05:17 up 4 days,  1:31,  1 user,  load average: 0.27, 1.59, 1.68
Tasks: 132 total,   3 running, 129 sleeping,   0 stopped,   0 zombie
Cpu0  :  5.2%us,  0.9%sy,  0.2%ni, 93.5%id,  0.1%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  5.5%us,  5.7%sy,  0.0%ni, 87.7%id,  0.9%wa,  0.0%hi,  0.1%si,  0.0%st
Mem:   2057456k total,   843060k used,  1214396k free,    30160k buffers
Swap:  4008208k total,    86208k used,  3922000k free,   318612k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 5703 feiticei  20   0  472m 159m 5296 S    9  7.9 896:35.31 ktorrent
22920 feiticei  20   0  491m  42m  11m S    3  2.1   1:02.21 amarokapp
 5085 root      20   0  746m  16m 2076 R    2  0.8  21:46.54 X
22989 feiticei  20   0  163m  57m 4228 S    1  2.9   0:34.28 wish
23771 root      20   0     0    0    0 S    1  0.0   0:00.09 pdflush
    1 root      20   0  3764  292  264 S    0  0.0   0:02.95 init
    2 root      15  -5     0    0    0 S    0  0.0   0:00.00 kthreadd
    3 root      RT  -5     0    0    0 S    0  0.0   0:00.16 migration/0
    4 root      15  -5     0    0    0 S    0  0.0   0:01.57 ksoftirqd/0
    5 root      RT  -5     0    0    0 S    0  0.0   0:00.23 migration/1
```

Processos

Processos: top

- Configurável para mostrar os processos por outras ordens

```
top d 1
top i
```

- Dentro do top:
  - <espaco> - Actualiza a informação imediatamente
  - h – ajuda do top
  - k – mata um processo
  - i – ou mostra processos zombie/parados
  - n – número de processos a mostrar
  - r – nova prioridade
  - 1 – mostra todos os processadores
  - f – campos a mostrar
  - F – ordem pela qual os processos são ordenados

Processos

Processos: Controlo de trabalhos

- Controlo de trabalhos foi inventado quando os utilizadores tinham apenas acesso a um terminal
- Permite aos utilizadores correr vários comandos em plano de fundo enquanto trabalham noutra aplicação em primeiro plano
- Quando um processo é posto em plano de fundo, chama-se um trabalho (job)
- A partir deste momento, não temos forma de lhe dar comandos (input)

Processos

Processos: Controlo de trabalhos

- Iniciar uma tarefa
  - vim
- Ctrl + z para suspender o programa
- Comando jobs para ver o status
- Enviar o trabalho #1 para o plano de fundo (background)
  - bg
- Iniciar outro programa, como top em plano de fundo
  - top &
- Correr o comando jobs

- Três designações:
  - + : indica o trabalho corrente e qualquer comando como fg ou bg actua naquele trabalho
  - - : indica o trabalho anterior ou o penúltimo onde iremos actuar
  - <nada>: indica um trabalho regular e nenhuma acção será efectuada nele

```
jobs
[1]+  Stopped                 vim
```

```
bg
[1]+ vim &
```

```
jobs
[1]-  Stopped   vim
[2]+ Stopped  top
```

Processos

Processos: Controlo de trabalhos

- Colocar o trabalho corrente em primeiro plano com fg
- O comando top deverá aparecer
- Sair do top com q
- Executar o comando jobs -l novamente para verificar que o vim é o trabalho corrente
- Matar o vim digitando o ID com o comando kill
- kill -KILL <PID>
- Correr novamente o comandos jobs e verificar

Curso Linux

bibliografia

- LPIC I, Exam Cram 2, Brunson - QUE Certification
- LPI Linux Certification In a Nutshell, Pritchard, Pessanha, Langfeldt, Stranger & Dean – O REILLY
- Linux Administration Handbook, Second edition, Nemeth Snyder Hein – Prentice Hall